Slirp is dead, long live Slirp!

A new approach to user-mode networking in qemu, and what KubeVirt can do with it

Stefano Brivio, Alona Paz

KVM Forum 2022

Who are we

Alona

- maintainer of KubeVirt, oVirt engine backend and oVirt UI
- with Container Native Virtualisation team at Red Hat
- actually knows how this stuff is used

Stefano

- on-and-off kernel developer (core networking, netfilter)
- with Virtual Networking team at Red Hat
- prefers pods stir fried, but also enjoys tofu and edamame

This talk is not about...

- Slirp (just a tiny bit)
- Rust (sorry!)
- microservices (could be worse)
- **The Cloud** (I swear)

This talk is about…

- Kubernetes
- KubeVirt
- existing KubeVirt networking
- why KubeVirt needed passt
- passt

Kubernetes

Kubernetes, also known as k8s, is an open-source system for automating deployment, scaling, and management of containerised applications.

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes. A pod is a group of containers, all the containers inside a pod share the same network namespace.

KubeVirt

KubeVirt is an add-on extending Kubernetes by adding resource types for virtual machines.

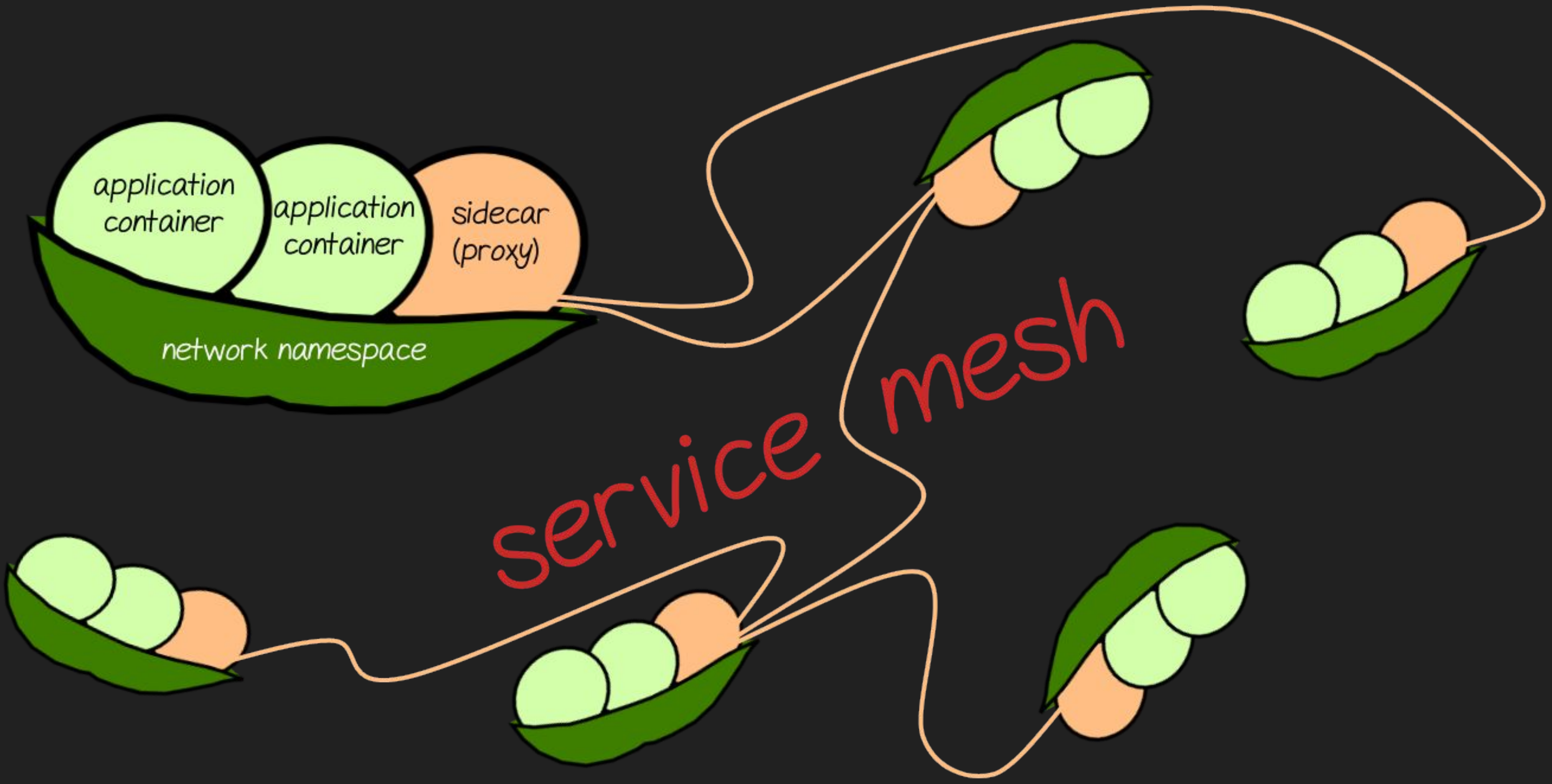
It allows users to run virtual machines alongside pods/containers in their Kubernetes clusters.

KubeVirt virtual machines run within regular Kubernetes pods, where they have access to standard pod networking, and can be managed using standard Kubernetes tools such as `kubectl`.

Service Meshes

A **service mesh**, like the open source project Istio, is a way to control how different parts of a system share data with one another.

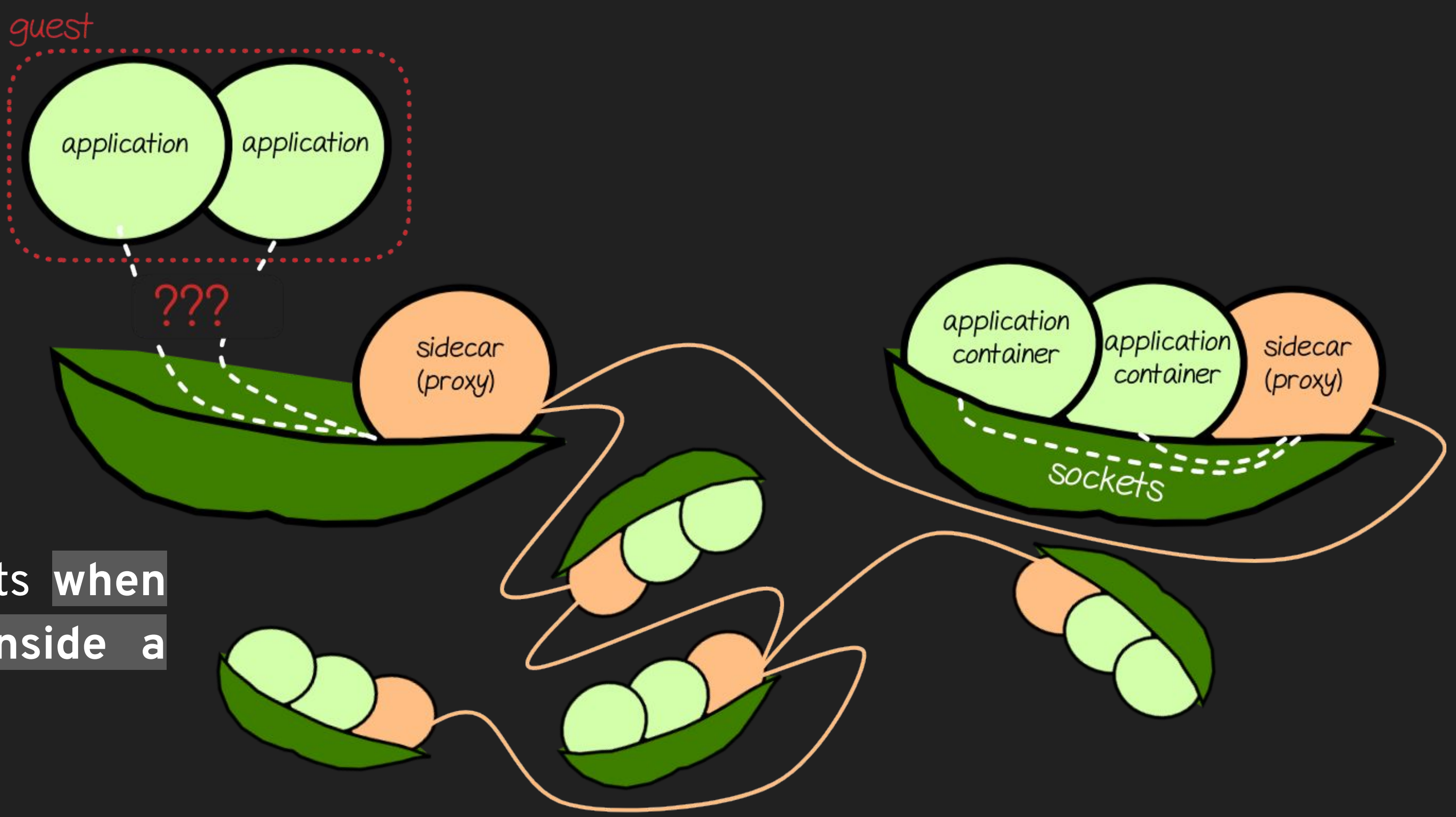
A **sidecar** (container) **proxy** sits alongside a microservice and routes requests to other proxies. Together, these sidecars form a mesh network.



Service Mesh Sidecar: Assumptions

- Expects applications to run on the **same network namespace** as the proxy
- Needs sockets and processes visibility for **socket redirection**, monitoring, **port mapping**
- Assumes that the proxy sees the Pod set of **addresses and routes**
- Assumes the application **traffic arrives from userspace**

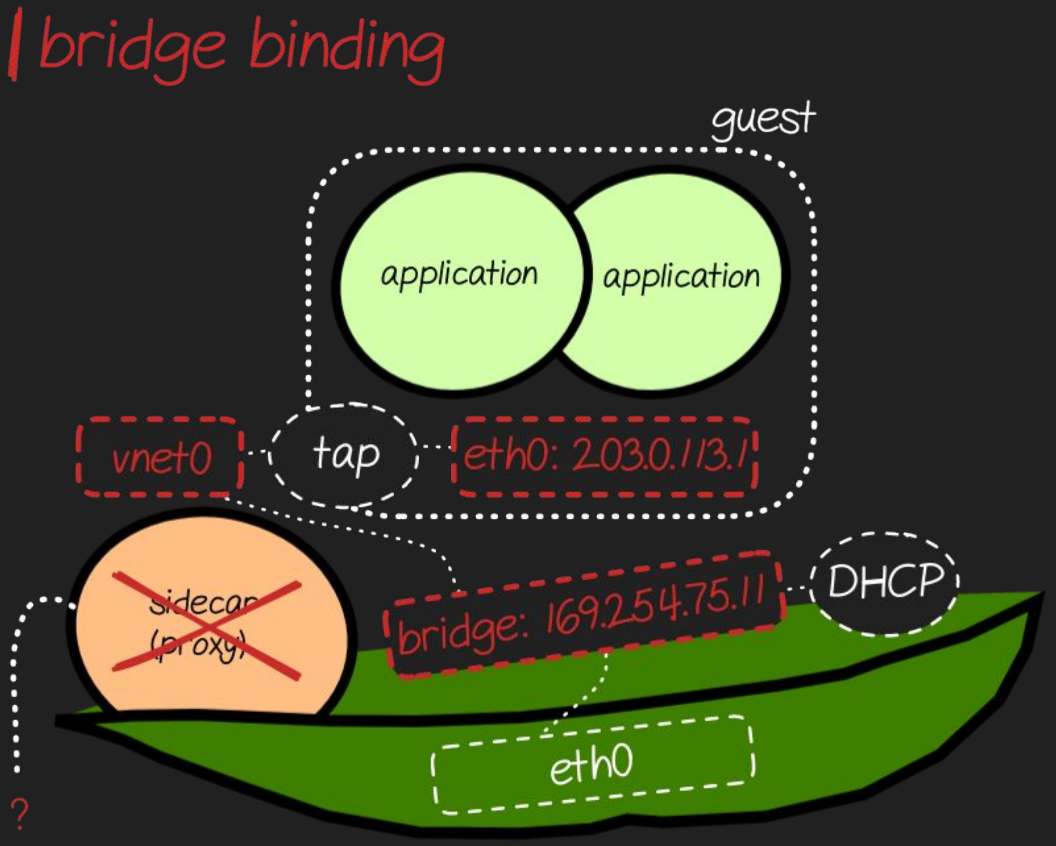
It is **hard to meet** those requirements **when the main application is running inside a virtual machine** and not a container.

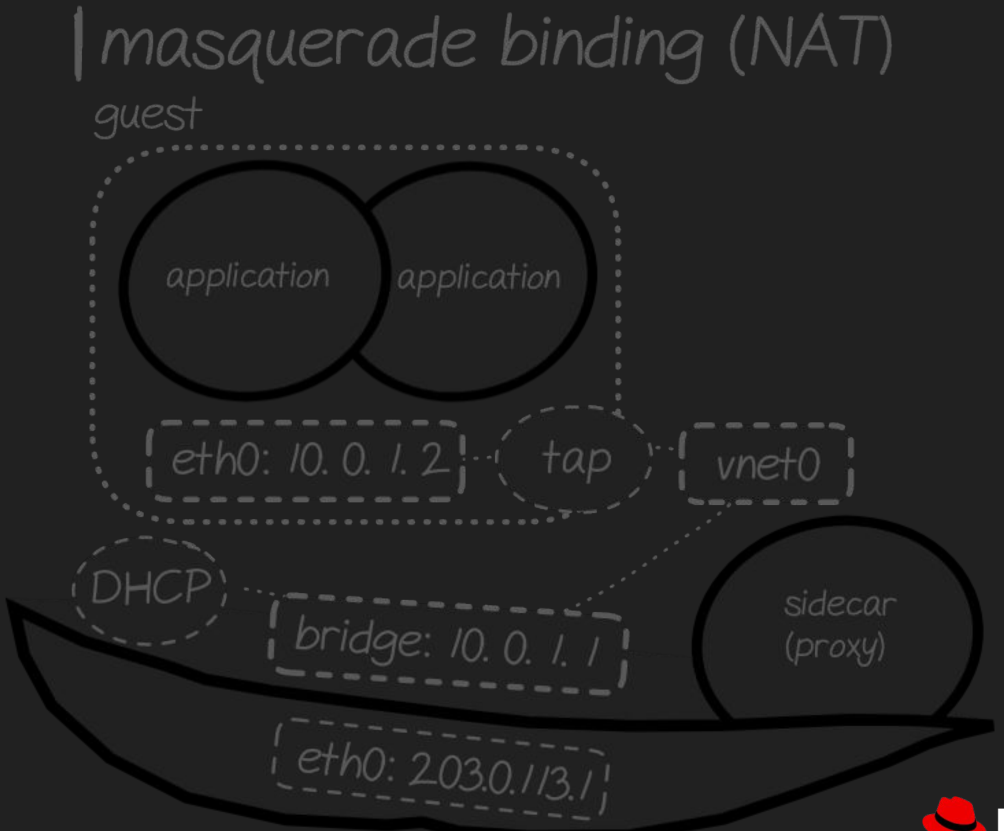


KubeVirt Networking: existing bindings

KubeVirt currently has two main options binding the virtual machine to the pod networking:

- Bridge binding - Stealing the pod's interface identity
- Masquerade binding - Using nftables rules to NAT the virtual machine traffic

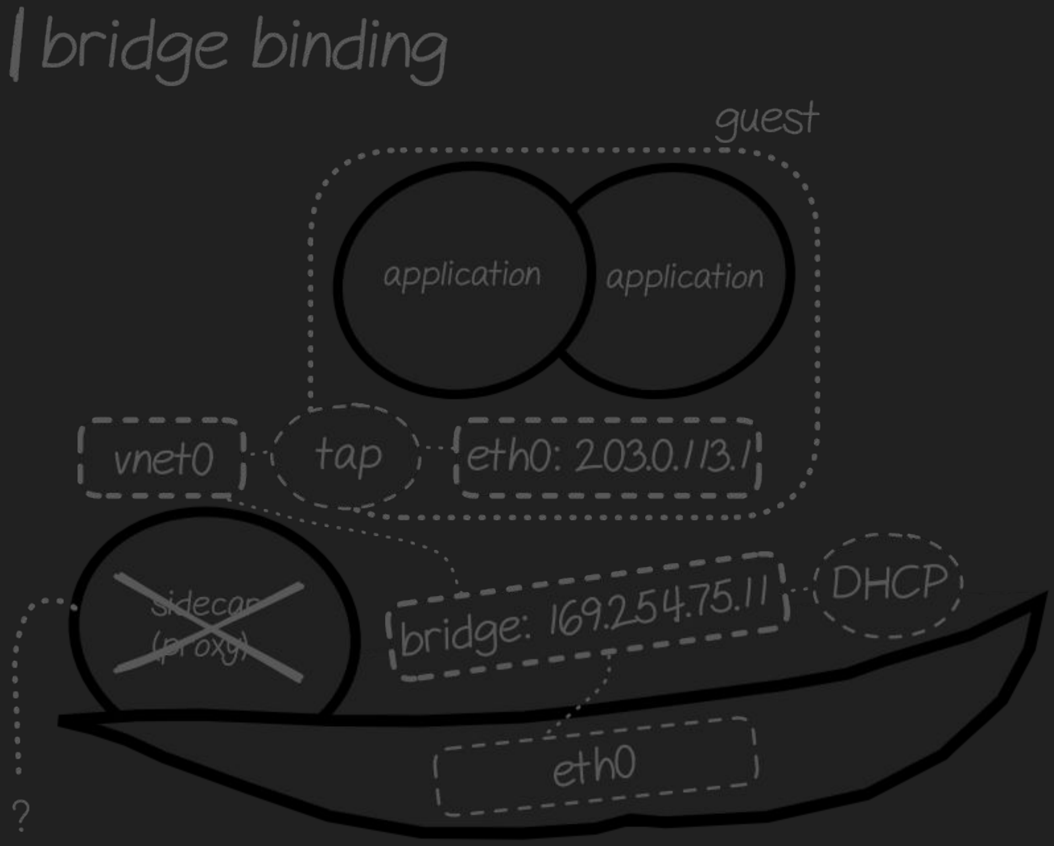


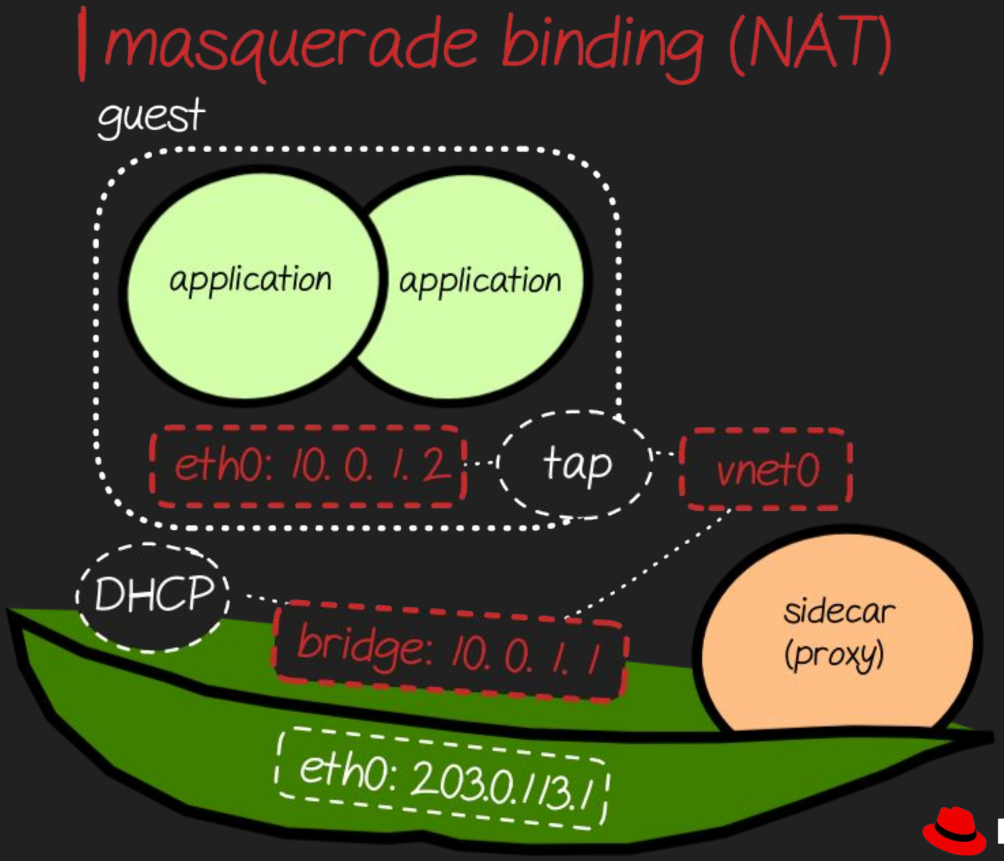


KubeVirt Networking: existing bindings

KubeVirt currently has two main options binding the virtual machine to the pod networking:

- Bridge binding - Stealing the pod’s interface identity
- Masquerade binding - Using nftables rules to NAT the virtual machine traffic





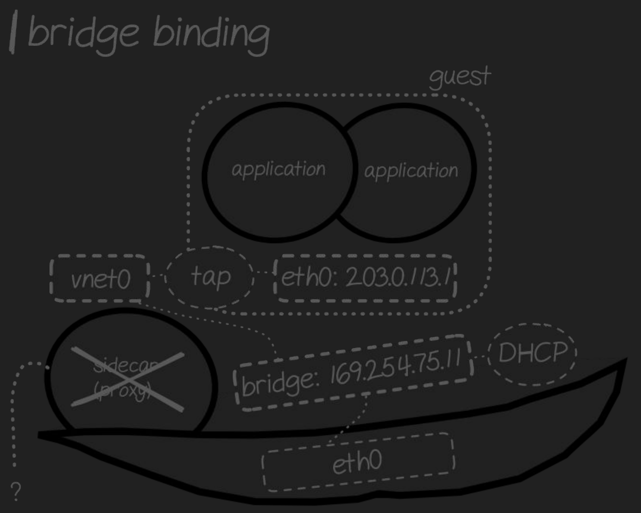
Downsides of bridge and masquerade bindings

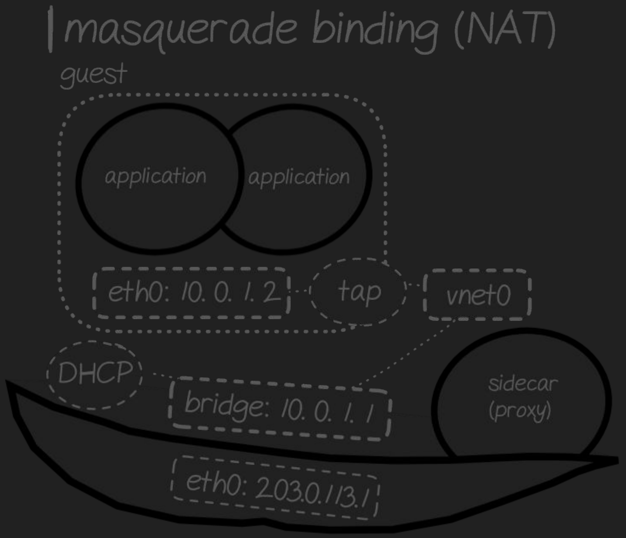
Both:

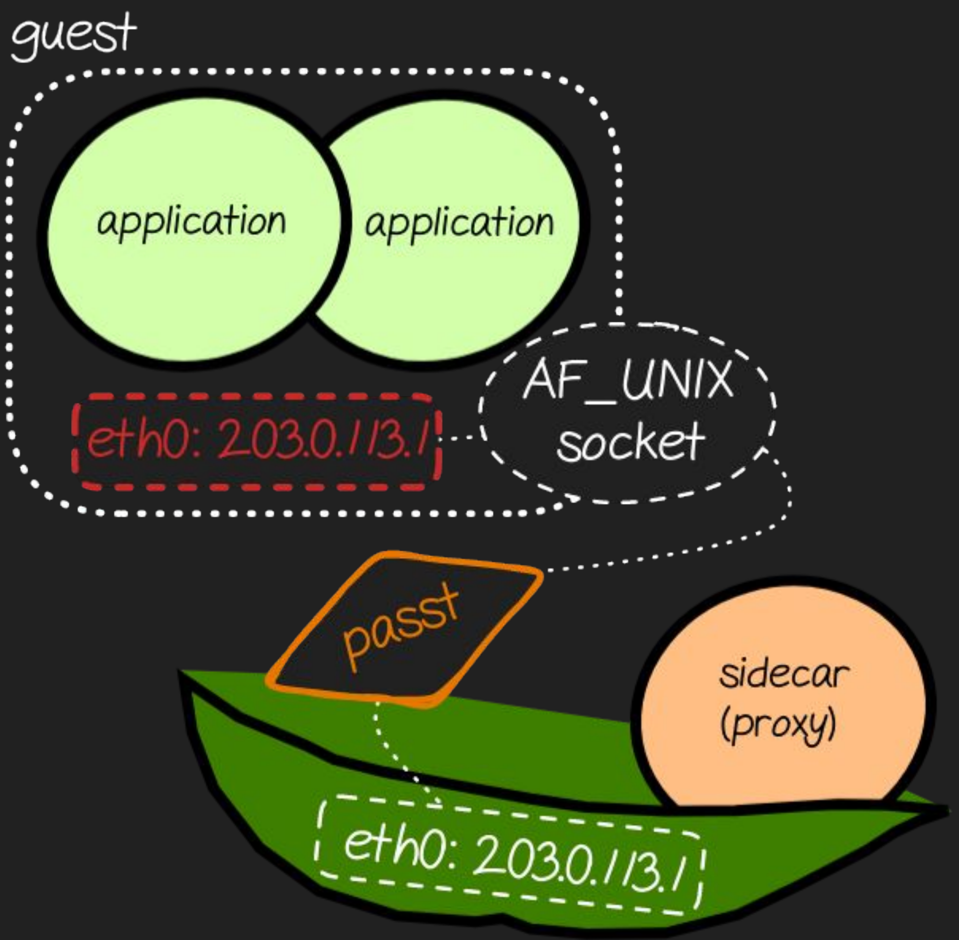
- ❌ don’t allow seamless integration with service meshes (bridge binding doesn’t allow sidecars, masquerade binding traffic doesn’t land in the pod’s userspace)
- ❌ had to be implemented by ourselves – introduce a DHCP server for them, and if we want multicast, we have to do that too
- ❌ require tricks to work on an unprivileged pod

KubeVirt Networking: passt binding

- passt binding - implements a translation layer between a Layer-2 network interface and native Layer-4 sockets







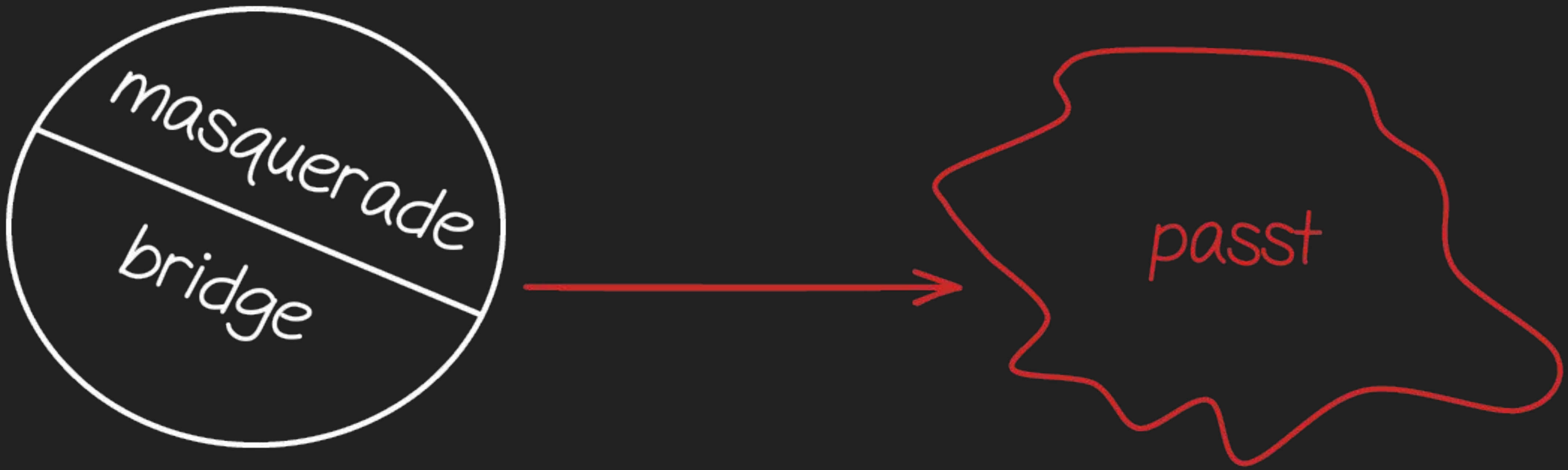
Advantages of passt as KubeVirt network binding

With passt:

- ✅ we can seamlessly integrate service meshes since it maps packets at Layer-2 to Layer-4
- ✅ we share a tool that is universal
- ✅ we don’t need extra networking security capabilities

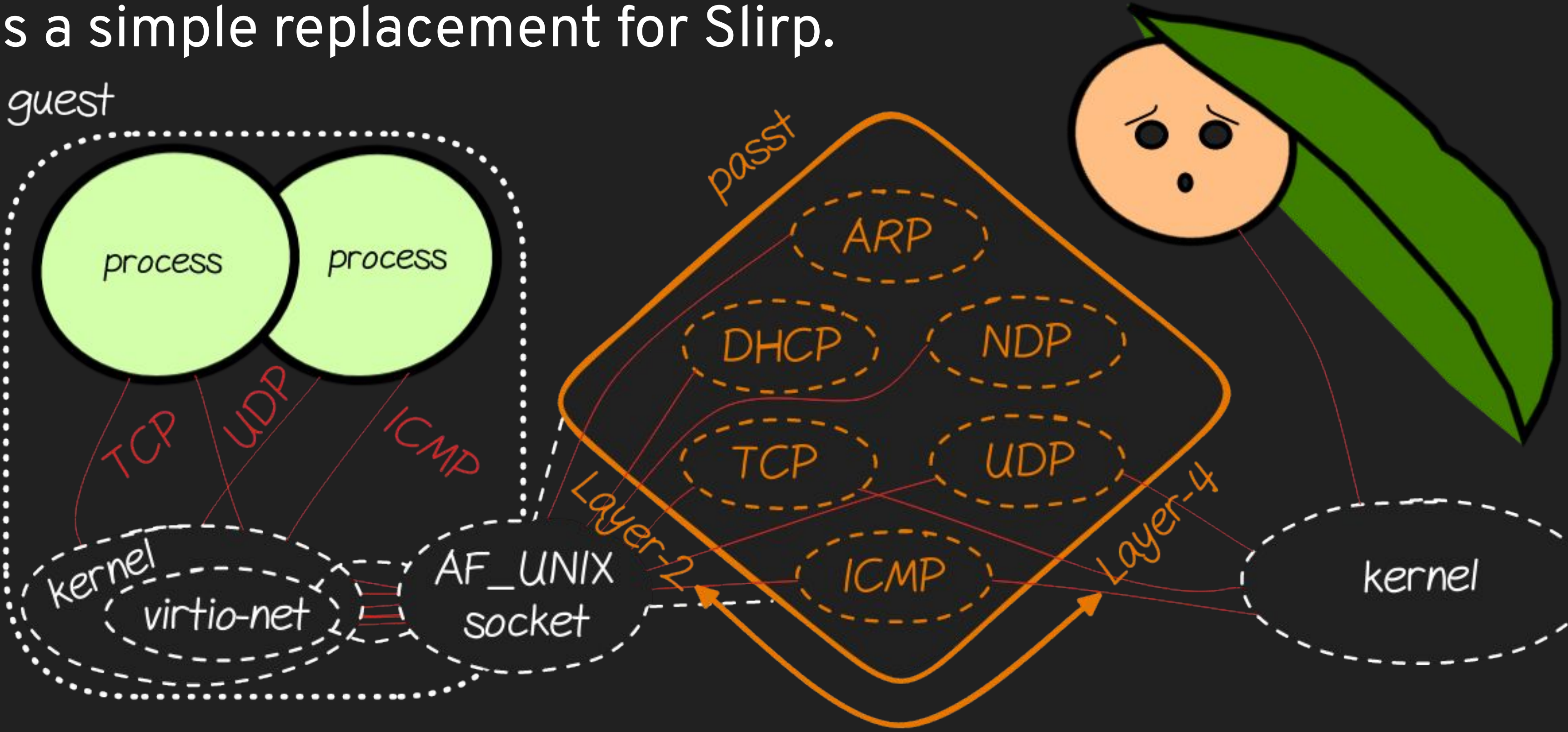
passt might be the future of KubeVirt networking

KubeVirt is thinking about substituting both existing bindings that are used to connect the Pod network with passt



Introducing *passt*

passt implements a translation between Ethernet frames from/to qemu (Layer-2, over AF_UNIX socket) and TCP, UDP, ICMP echo sockets (Layer-4). It doesn’t require any capabilities or privileges, and it can be used as a simple replacement for Slirp.



“Like Slirp, but not Slirp”
[anonymous KubeVirt developer, 2020]

Why not Slirp?

Slirp also maps traffic between qemu interface and host sockets, without the need for security privileges. However, it was developed for a different purpose:

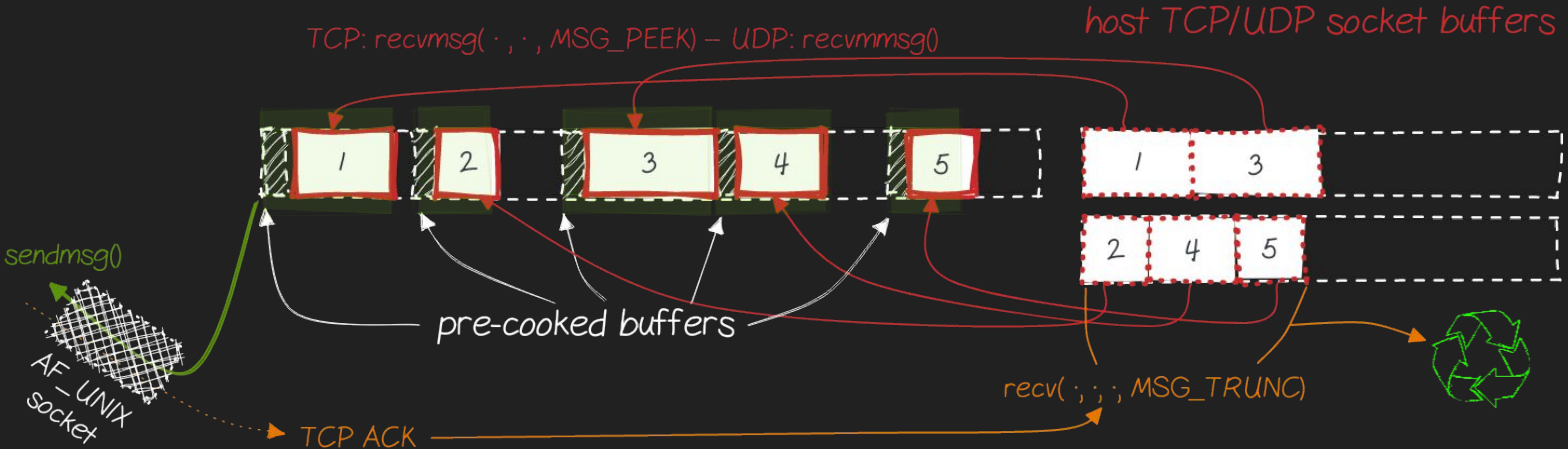
- no focus on performance – no TCP window scaling (!), no syscall batching
- forces NAT, no way to copy addressing/routing from host (convenient for KubeVirt)
- implements a full TCP stack, not bare essential
- offers only partial IPv6 support: no NDP, no DHCPv6, no port mapping to guest
- runs in the same process context as qemu
- ships with a number of features such as application-level gateways (FTP, IRC), SMB support, etc. which contribute to increase the attack surface

Red Hat

Back to passt: data path, host to guest

passt doesn't keep per-connection buffers. This avoids the need for dynamic memory allocation, but it means that we can't dequeue data until segments are acknowledged.

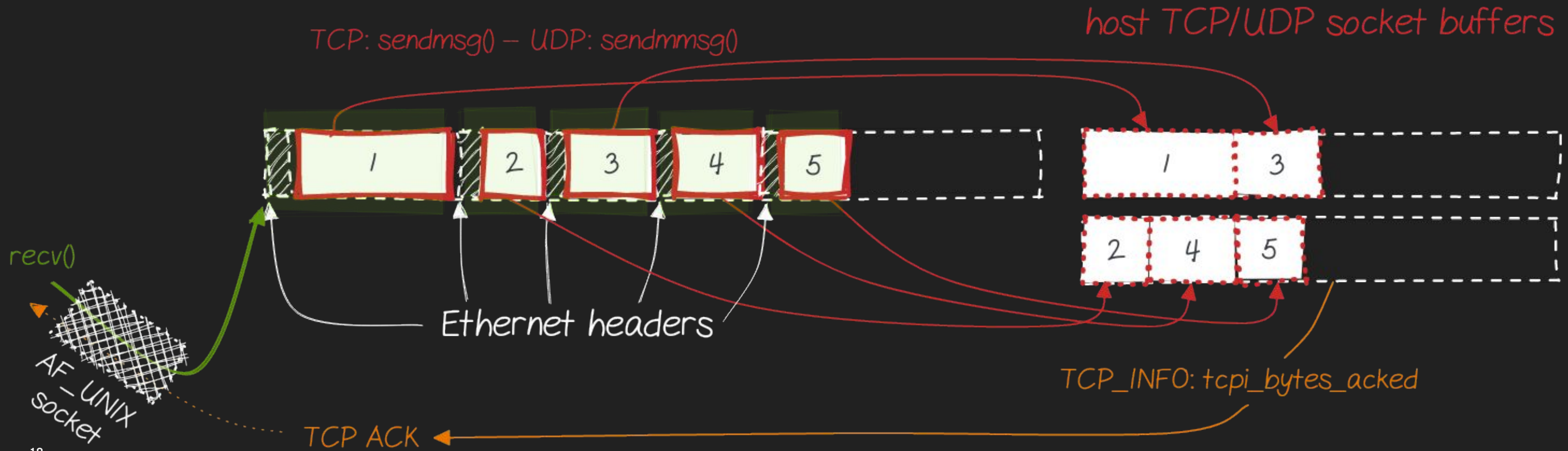
Read socket data into pre-cooked buffers and send batches to qemu, keeping the amount of syscalls to the bare minimum.


TCP: recvmsg(· , · , MSG_PEEK) – UDP: recvmmsg()
host TCP/UDP socket buffers
1
2
3
4
5
1
3
2
4
5
sendmsg()
AF_UNIX socket
pre-cooked buffers
recv(·, ·, ·, MSG_TRUNC)
TCP ACK

Data path: guest to host

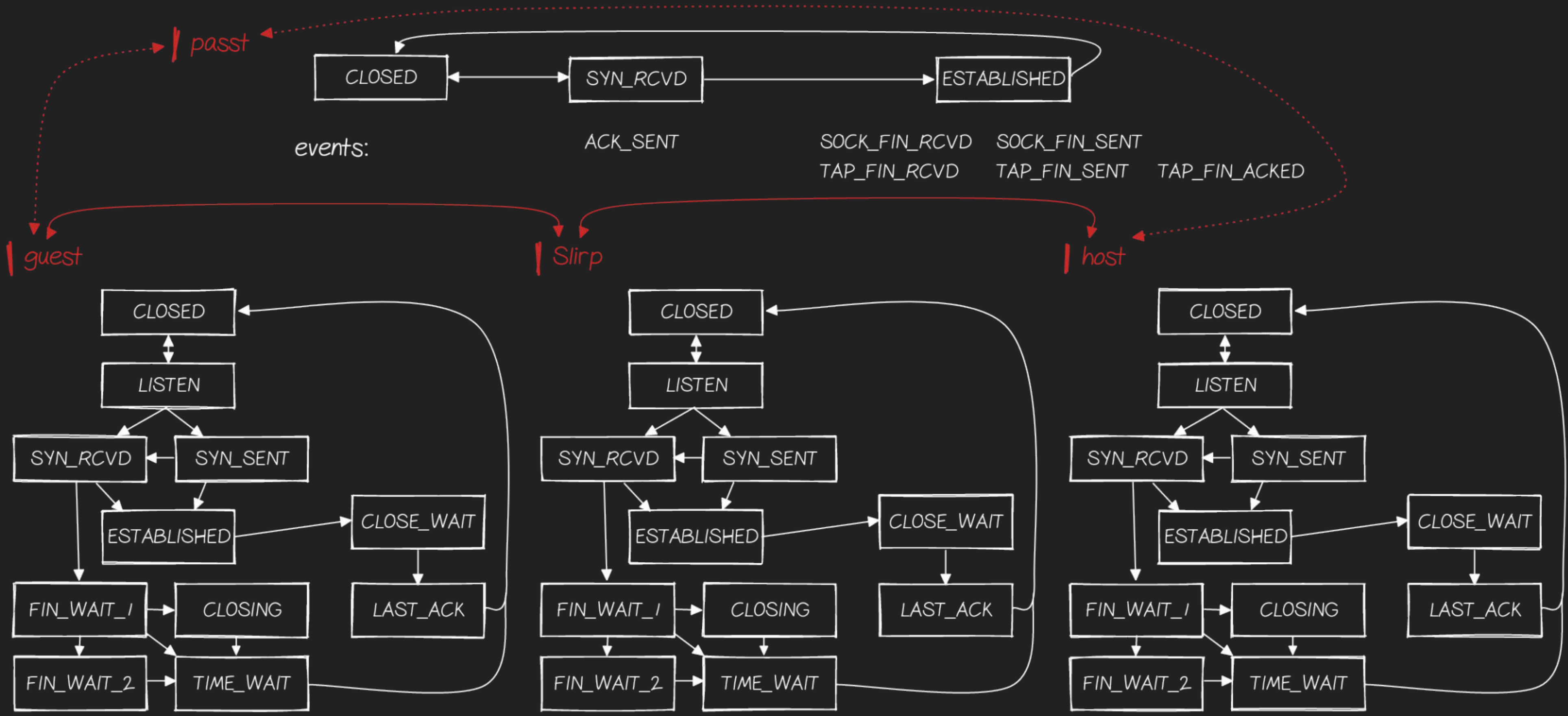
Read packets from qemu into single buffer, no further copies, scan headers and queue to sockets.

Send acknowledgement to the guest when the receiver sends them, delegating congestion control to original sender and receiver.


TCP: sendmsg() – UDP: sendmmsg()
host TCP/UDP socket buffers
1
2
3
4
5
1
3
2
4
5
recv()
Ethernet headers
AF_UNIX socket
TCP_INFO: tcpi_bytes_acked
TCP ACK

TCP Adaptation

No full TCP state machine, three states only: the host kernel already implements a number of them.
Event flags define possible transitions.



Security Topics

- mount, PID, IPC and UTS namespaces are unshared (sandboxing)
    - no access to filesystems or other processes after initialisation
- no capabilities needed, as it doesn't create network interfaces (`CAP_NET_ADMIN`)
- won't run as root
- no dynamic memory allocation: sbrk(2), brk(2), mmap(2) blocked by seccomp
- strict seccomp profiles: only 26 syscalls allowed on x86_64
- no external dependencies other than a standard C library
- SELinux and AppArmor profiles shipped with repository

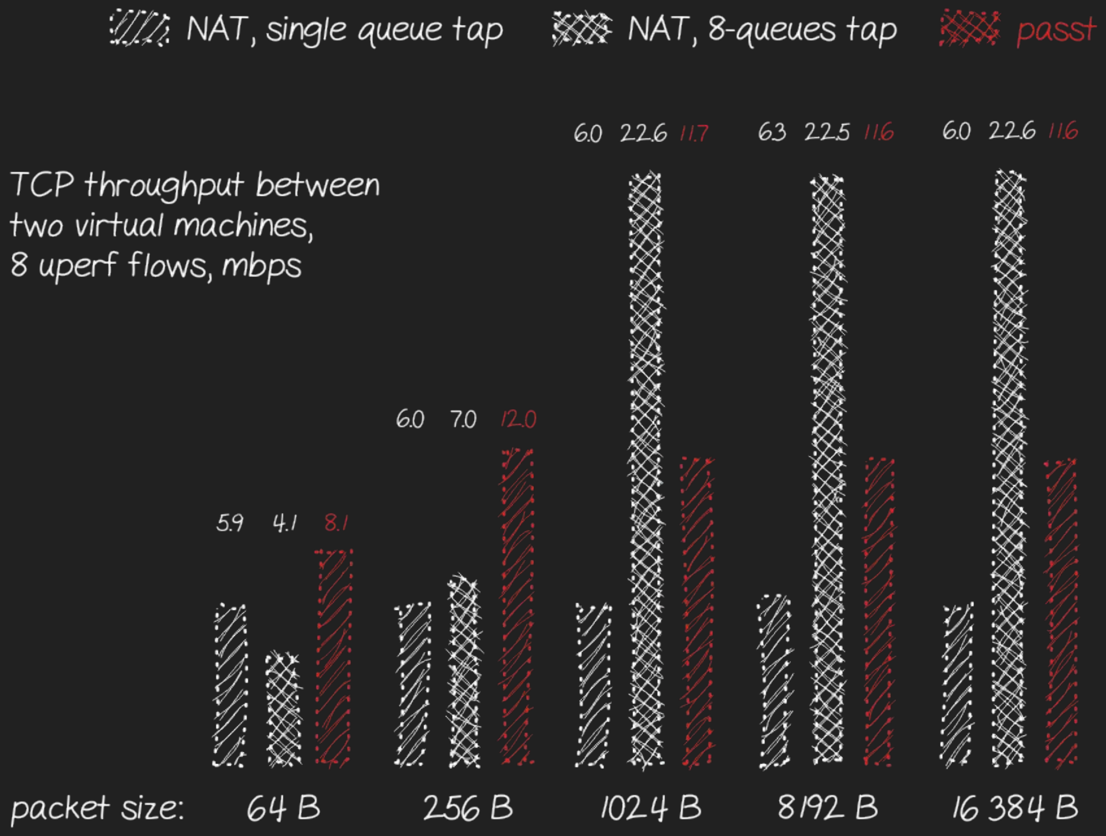
- *not* written in Rust (at least for the moment): that would exclude some classes of vulnerabilities (stack-based overflows), but makes it hard to forbid dynamic memory allocation
    - It should be possible to use only the Rust Core library inside the main loop
    - …but buffer alignment requirements would still require a heap of `unsafe` code

Performance Topics

- the guest can use a 64 KiB MTU: no *packets* to the host kernel, which deals with segmentation
- segments are coalesced, messages are batched (including ACK segments)
- AVX2 checksum routines (on x86_64)
- pre-cooked buffers help with data locality
- two cache-hot copies, on any path
- no additional congestion control
- no userspace overhead: it's all syscalls
- still a gap with multi-queue tap

Plan: add vhost-user back-end:

- one copy instead of two from qemu to host kernel
- no AF_UNIX copy pair (albeit cheap)



```
git clone git://passt.top/passt && make -C passt
```

Availability and Integrations

- Linux only, for the moment
  - TCP_INFO socket option limits portability. Some BSD flavours now have equivalents
- Fedora packages available, unofficial packages for Debian, Ubuntu, OpenSUSE, CentOS
  - Try it now: https://passt.top/#try-it, check automated demos and CI
- KubeVirt tech preview, PoC for Kata Containers
- qemu integration: currently using a wrapper for AF_UNIX socket, series by Laurent Vivier for native support pending merge (hopefully not anymore as we speak!)
- libvirt: out-of-tree patch available (still using wrapper), updated draft in progress

contribute: lists, bugs, chat

pasta: same binary, for namespaces (slirp4netns equivalent)

- Podman integration proposed, pending wide package availability

Credits

David Gibson (test framework, namespace options, etc.)
Fabian Deutsch (sponsoring the idea)
Jenifer Abrams (performance testing)
Laurent Vivier (AF_UNIX socket support in qemu)
Maya and Lenny (cat and rabbit who wrote most of passt’s codebase)
…and many others.

THE END

I mean, questions?